Jabba: Hybrid Error Correction for Long Sequencing Reads using Maximal Exact Matches

Giles Miclotte, Mahdi Heydari, Piet Demeester,
Pieter Audenaert, and Jan Fostier*

Ghent University - iMinds, Department of Information Technology,
Internet Based Communication Networks and Services (IBCN)
Gaston Crommenlaan 8 (bus 201), B-9050 Gent, Belgium
{giles.miclotte,mahdi.heydari,piet.demeester,
pieter.audenaert,jan.fostier}@intec.ugent.be
www.ibcn.intec.ugent.be

Abstract. Third generation sequencing platforms produce longer reads with higher error rates than second generation sequencing technologies. While the improved read length can provide useful information for downstream analysis, underlying algorithms are challenged by the high error rate. Error correction methods in which accurate short reads are used to correct noisy long reads appear to be attractive to generate high-quality long reads. Methods that align short reads to long reads do not optimally use the information contained in the second generation data, and suffer from large runtimes. Recently, a new hybrid error correcting method has been proposed, where the second generation data is first assembled into a de Bruijn graph, on which the long reads are then aligned. In this context we present Jabba, a hybrid method to correct long third generation reads by mapping them on a corrected de Bruijn graph that was constructed from second generation data. Unique to our method is that this mapping is constructed with a seed and extend methodology, using maximal exact matches as seeds. In addition to benchmark results, certain theoretical results concerning the possibilities and limitations of the use of maximal exact matches in the context of third generation reads are presented.

Keywords: sequence analysis, error correction, de Bruijn graph, maximal exact matches

1 Introduction

The accurate determination of the DNA sequence of an organism, i.e., establishing the precise order of the nucleotides A, C, G and T in a DNA molecule, is a fundamental and challenging problem in biology. Essentially this process consists of two steps: (i) sequencing the DNA by means of a chemical process, resulting in a large number of reads and (ii) genome assembly, where the reads are processed to reconstruct the complete DNA sequence. Every sequencing technology results

* To whom correspondence should be addressed.

in reads that contain errors, with error profiles varying greatly between platforms. There is a clear distinction between *second generation* reads and *third generation* reads, where the latter are characterized by vastly improved read lengths albeit with much higher error rates. Processing such reads usually involves mapping them to other sequences, either by aligning the reads to each other to establish potential overlap, or by mapping them to a reference genome. Errors in the reads introduce noise to these alignments, leading to weaker alignments than the corresponding error-free reads would have. Lower rated alignments may then be discarded for further analysis, potentially discarding crucial information. This can be especially problematic when dealing with low quality reads in a region with low coverage. To deal with this sequencing noise, error correction methods can be applied. By correcting the errors in the reads, the optimal alignments can be more accurately identified and more appropriately rated, leading to better downstream analysis, as shown in e.g. [1] for *de novo* assembly.
In Jabba third generation reads are aligned to a de Bruijn graph built from second generation reads, using a seed-and-extend approach. The resulting paths in the graph dictate the read correction. The seeds are maximal exact matches (MEM) between an individual read and a node of the graph. The usage of MEMs as seeds has several advantages over k-mers. Firstly, the seeds can be longer. Even though long seed occur only rarely, few longer seed can be sufficient to have a rough estimate of how the read should be aligned to the graph. Shorter seeds can then be used to further refine this. Secondly, given an enhanced suffix array, seeds of arbitrary lengths can be sought without the need to rebuild this index. This is not the case for a k-mer index (e.g. a hash table). In case different values for k are to be used during the alignment process, different k-mer indexes need to be build. Finally the MEMs are not required to have the same size as the nodes. Since the high error rates of the third generation reads are the limiting factor on the minimal seed size, this allows the use of a larger value of k to build the de Bruijn graph, resulting in a less complex de Bruijn graph.

1.1 Related work

For second generation sequencing we mainly consider Illumina. The different Illumina technologies produce many short (100-250 nucleotides) reads with a high accuracy (<2% errors, mainly substitutions) with high throughput and at a low financial cost. New algorithms, based on de Bruijn graphs, were specifically developed to efficiently deal with the assembly of huge amounts of second generation sequencing data. Overlap between short reads is then established in linear time between reads that share a k-mer, i.e., a substring of length k.

Algorithms to correct second generation reads have been classified [2] into three types. The k-mer spectrum-based methods rely on coverage thresholds to determine whether a k-mer represents part of the actual DNA sequence, these methods have been used for second generation error correction [3, 4], but also in *de novo* genome assembly algorithms [5] and hybrid error correction methods [6].

The suffix tree-based methods [7, 8] generalize the k-spectrum methods by handling multiple k values at once, while the multiple sequence alignment-based methods [9] correct the reads after aligning several similar reads.

Recently, third generation sequencing technologies (Pacific Biosciences, 2013; Oxford Nano Technologies, 2014) began to emerge. Pacific Biosciences SMRT sequencing results in much longer reads (avg. >5000 nucleotides), albeit with significantly higher error rates (15%, mostly insertions and deletions and to a lesser extent substitutions). Despite this high error rate, the errors are uniformly distributed over the read, leading to a very high consensus accuracy if the coverage is sufficiently high and overlap between the reads is correctly established. Computing such overlap can not be efficiently achieved by means of a de Bruijn graph, because the high error rate leads to an overabundance of incorrect k-mers. Other efficient methods have been developed to compute pairwise alignments between third generation reads [10, 11].However, the coverage required for high accuracy consensus-based correction can still lead to a prohibitively high financial cost for many sequencing projects.

From this perspective, the use of hybrid error correction methods appears to be an attractive alternative. The goal is to correct long third generation reads using the more accurate sequence information contained in second generation reads. The idea is that a sufficient coverage in (cheap) second generation data might be sufficient to correct the long reads, regardless of the coverage of third generation data. This may result in a reduced financial cost for sequencing as low coverage third generation data might suffice. It should be noted that lower third generation coverage can directly result in lower assembly quality, no matter the quality of the reads, because of the uneven length distribution of the reads [12]. However, hybrid error correction methods also appear attractive from a computational point of view as they avoid pairwise comparisons between long reads, thus circumventing the quadratic computational complexity. The first type of hybrid error correction methods [13–15] rely on mapping short reads to long reads, and then calling the consensus sequence from this multiple alignment. However, such methods map short reads individually and do not exploit the context in which the short read occurs. A newer hybrid error correction method, LoRDEC [6], first constructs a de Bruijn graph from the short reads and then maps the long reads on this graph. The sequence implied by the path in the graph to which the long read aligns then represents the corrected read. The use of a de Bruijn graph has the advantage that overlap between short reads is established prior to mapping them to long reads. In [6], it was shown that LoRDEC achieves similar accuracy as other error correction methods, but with significantly improved runtimes.

2 Methods

2.1 Overview

In this work, we further build upon the idea of using a de Bruijn graph for hybrid error correction of long reads. Specifically, our main goal is the use of

Illumina data to correct Pacific Biosciences SMRT reads. A de Bruijn graph is then constructed from Illumina data and corrected using standard procedures (see further). Subsequently, long reads are aligned along a certain path in the graph in order to correct them. Whereas LoRDEC relies on shared k-mers to align long reads to a de Bruijn graph, we explore the idea of using maximal exact matches (MEMs). MEMs are exact matches between two sequences that can not be extended in either direction, this as opposed to common k-mers, which are exact matches of a fixed length k, which may or may not be extendable. Alignment methods based on maximal exact matches have been developed for read mapping [16–18]. It is shown in [16] that these methods can be more efficient than alignment techniques based on k-mers and Burrows-Wheeler transforms [19, 20]. From the definition of a MEM, it is clear that every MEM of size $l \geq k$ can be represented as a consecutive sequence of k-mers, and vice versa. As such, finding and storing MEMs can be achieved in a more efficient manner, since MEMs can compactly represent multiple k-mers. The remainder of this section is dedicated to a more in-depth description of all steps involved (Fig. 1).

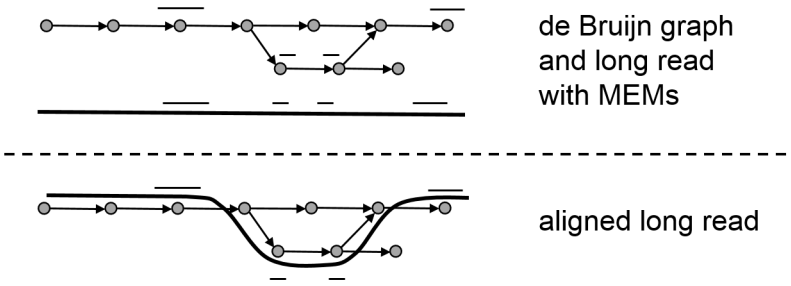

Fig. 1. To align a read to the de Bruijn graph, a seed-and-extend algorithm is used. First MEMs are found between the read and the graph, then a path in the graph is found between these seeds, creating the final alignment.

2.2 Correction of the de Bruijn graph

Errors in short reads lead to erroneous paths in the de Bruijn graph. The errors in the graph can be corrected as described in [5]. Two types of errors can be discerned based on their position in the read. An error that is located at least $k-1$ nucleotides away from both ends of the read will lead to k erroneous k-mers. In turn, this leads to the formation of a 'bubble', i.e. a path of length k that runs parallel to the real path. Assuming a sufficiently low error rate and a high coverage the correct path will typically have a higher coverage than the parallel erroneous paths, and the graph can be corrected by removing the erroneous path. On the other hand, errors positioned close to the ends of the read lead to the creation of less than k erroneous k-mers, thus forming 'dead ends' (tips) in the de Bruijn graph. These can be identified and removed based

on topology and coverage considerations. Errors in the reads may also result in erroneous connections between unrelated parts of the graph, and because of coverage biases certain paths could be absent or underrepresented in the graph. This vastly complicates the graph correction procedure and erroneous paths may remain present in the final corrected graph.

2.3 Aligning reads to a de Bruijn graph

To align the reads to the graph a seed-and-extend approach is applied. By properly indexing the graph the seeds can be found in $O(m)$ time, where m is the size of the read that is being mapped.

Finding Maximal Exact Matches To rapidly find MEMs between the nodes of the graph and the long reads, essaMEM [21] is used. These MEMs will be used as seeds for our alignment. By concatenating the sequences of every node and their reverse complement, a sequence is constructed. From this sequence an enhanced sparse suffix array is built by essaMEM. The sparseness factor of the index sharply reduces the space requirement for the index, compared to traditional suffix trees or enhanced suffix arrays, but this comes at the cost of a small increase in runtime. The required space could be even further reduced by only indexing the nodes and not their reverse complements. This would however double the search time, since the reverse complements of the reads then also have to be matched.

Chaining Seeds To chain the seeds several passes over the read are performed. In each iteration the algorithm considers every region of the read that has not yet been corrected. For every such region separately, the largest seeds are considered. From these seeds it is determined to which nodes the current region of the read could map. For each such node the list of all seeds between this node and the current region of the read is considered, and an optimal placement of these seeds is decided, removing the ones that do not fit. Seeds are compatible if the distance between the two seeds on the read is contained in an interval determined by the estimated error rates and the distance of the seeds in the node.

Generally larger MEMs are less likely to be noise than shorter seeds, since the number of all k-mers increases exponentially if k increases and the number of k-mers contained in a sequence is similar to the size of the sequence, independent of k. There can still be noisy long seeds, especially when the genome contains imperfect repeats. In this case, the correct seeds can usually be recognized amidst the noisy seeds by considering the context. Firstly, the local context is considered, by comparing the seeds in the same node. This way seeds that occur in the same order in a node and in the read can be chained together to form inexact matches. Secondly, if the situation is still ambiguous, the global context is considered, by comparing the alignments in the neighborhood of the ambiguous region. If this neighborhood has not yet been chained in previous passes, the chaining of the current region is delayed to the next pass.

After obtaining the presumed layout of the seeds, the quality of the alignment is checked. Large gaps or a relatively large amount of mismatches may indicate incorrect alignments. The following cases are filtered:

1. Local mappings that are not super maximal, i.e., local mappings that are on the read contained in a larger local mapping.
2. Local mappings that cover less than 20% of a node. The absence of any seeds in the rest of the node makes it less likely that this is actually a correct mapping.
3. Local mappings to nodes of size smaller than half of the largest local mapping. It is preferred to extend from those larger local mappings, since those are more reliable.

These filters are linearly relaxed with each pass of the algorithm, in the last iteration all local mappings are considered. After the local alignments are computed for the current pass, the next phase begins: chaining the alignments between different nodes by following unique paths in the graph. During this phase every local alignment is extended by considering the possible paths in the graphs. Both directions of the alignments are extended in the same manner, as follows:

1. If there is a unique edge, this edge must be correct and the local alignment is extended along this edge.
2. If there are several edges, the lengths of the end nodes are considered. Since the extension takes place between two regions of the read, certain estimates can be made for the maximal distance between the alignments, edges that are too long are then not considered.
3. If at any point there are no suitable edges to extend along, a mistake was made at some point. Either the graph is incorrect or the original local chaining was erroneous. In either case the erroneous region is reprocessed in a new local chaining step.

In the rest of this section the distance between corrected regions on a read is denoted as n and the estimated insertion and deletion rates of the data are denoted as i and d. After the unique-extension step, the resulting chains may overlap in the graph, in which case they can be linked together to make one consecutive path. Overlapping chains are however not a sufficient condition for linking, the sequences represented by the path and the read need to be compared. If the sequence on the path is smaller than $(1 + 2i)^{-1}n$, the shortest cycle at the common point is considered. If this shortest cycle can not adequately fill the gap, then the paths are not joined and the gap is left for the next pass. To determine whether the shortest cycle is a good fit a local alignment is performed between the sequence dictated by the path and the read, using a Smith-Waterman approach [22]. Likewise, if the resulting chains do not meet, the shortest path between both end points is considered. If this shortest path can not adequately fill the gap, the gap is again left for the next iteration.
By building from seeds and only using shortest path algorithms to chain the nodes, computationally expensive path searching can be avoided, however, this

can not be avoided indefinitely. After the final iteration a bounded path search is performed between consecutive corrected regions, in an attempt to fill the remaining gaps. This search looks for paths that contain a sequence with length bounded by the interval $[(1+2i)^{-1}n, (1+2d)n]$.

Read Ends The parts of the read beyond the extremal seeds, i.e., the ends of the read, are not corrected. Such a correction can be trivial, if the end is completely contained within the same node as the extremal seed, or in the unique path flowing from that node. In this case correcting the end would not provide any information about the genome that is not already contained in the graph. The correction can also be far from trivial, if there are several possible paths. In this case the correction of the end requires aligning the possible paths against the read. This can be done by looking for seeds with a lower threshold, or by performing direct global alignments with a dynamic programming approach. This is not done in Jabba, since it is a relatively expensive operation for a small gain.

Settings Jabba takes several parameters that can affect the results. Most importantly the minimal length l of MEMs for the initial search can be specified, the standard value is $l = 20$, but this should be chosen based on the discussion in section 3.1 in function of the data. The maximal number p of iterations of the algorithm can be specified, the standard value is $p = 5$. A third parameter allows Jabba to trim the reads beyond the extremal corrected positions.

3 Results concerning Maximal Exact Matches

In this section the occurrence of maximal exact matches in reads is investigated. Insertions and deletions have a different effect on the size of maximal exact matches than substitutions. A substitution error puts a firm stop to any running exact matches, while an insertion or deletion may allow for the exact match to continue, effectively looking like an error at a further point in the read. In the following this difference is ignored and all errors are treated like they were substitutions. Because of this, the size of MEMs is slightly underestimated for sequences that contain insertions or deletions. It is also assumed that errors are uniformly distributed in the sequences, as is the case for Pacific Biosciences SMRT reads.

3.1 Coverage by exact regions

In this section the expected ratio of a long read that should be covered by MEMs larger than a given size is explored, under the assumption that the reference contains no errors. Variations on this topic have been explored in [23–25]. In the following n is the length of the read, p is the error-rate and m the threshold for maximal exact matches. An *exact region of size* k on a read is defined as k

correct consecutive bases in that read. The *coverage by exact regions* is the ratio of bases that are contained in exact regions.
The expected number of exact regions (including those of length 0) is the expected number of errors, i.e., np. The expected coverage of a read by exact regions of size k is then the product of (i) the coverage of the read by one exact region of size k: k/n, (ii) the expected number of exact regions: np, and (iii) the probability that an exact region has size k: $(1-p)^k p$. This results in:

$$k(1-p)^k p^2 \ . \tag{1}$$

Summing (1) over all $k \geq m$ gives the expected coverage of the read by exact regions of size $k \geq m$:

$$\sum_{k=m}^{\infty} k(1-p)^k p^2 = (1-p) - \sum_{k=0}^{m-1} k(1-p)^k p^2 \ , \tag{2}$$

the right hand side provides a finite formula to compute this expected coverage. Figure 2 shows the expected coverage by exact regions larger than m, for error-rate $p = 15\%$ and $p = 35\%$. The maximum $1-p$ is obtained at $\{0, 1\}$ since every correct base is contained in an exact region of size ≥ 1. It can be seen that increasing p leads to a steeper descent near the inflection point. While it was a priori clear that a lower error rate leads to larger exact regions, this also shows that the equilibrium between a sufficient amount of seeds and a sufficiently large minimal seed length, is less stable for higher error rates.

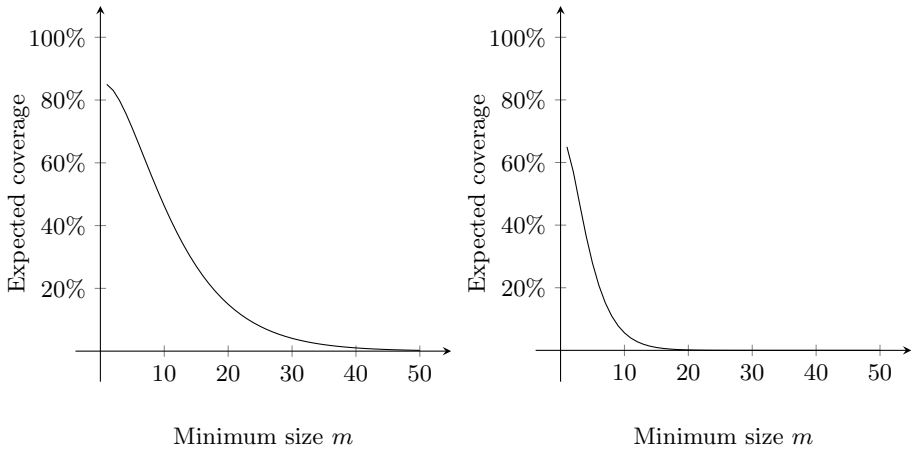

Fig. 2. Expected coverage by exact regions of size $k \geq m$ for reads of size 10000 with 15% errors (*left*) and 35% errors (*right*), expressed as percentages of the whole read as a function of the minimal size of the exact regions.

3.2 Occurrence of Exact Regions

The expected length of the longest exact region in a read of size n is denoted by $ER_p(n)$. If $np(1-p)^m \geq 1$ then at least one exact region of size $k \geq m$ is expected in a read of size n, hence the expected length of the longest run can be approximated by solving $np(1-p)^m = 1$ for m:

$$ER_p(n) \approx -\log_{1-p} np. \tag{3}$$

The distribution around this average can be approximated by the complement of a Gumbel distribution with cumulative distribution function

$$F(x) = \exp -(1-p)^{x+1}; \tag{4}$$

the probability that a read of length n will have an exact region of size $k \geq m$ is then approximated by

$$P(n,p,m) = 1 - F(m + ER_p(n)) = 1 - \exp\left(-np(1-p)^{m+1}\right). \tag{5}$$

These approximations are highly accurate when p and n are sufficiently large. Figure 3 shows the ratio of reads of length n that are expected to have an exact region of size m. For sufficiently large values of n, replacing n by $n' > n$ shifts the graph to the right by a term $\log_{1-p} n/n'$, replacing p by $p' < p$ shifts the graph to the left and steepens the descent near the inflection point. This again shows that larger error rates make the determination of a proper seed size threshold less stable.

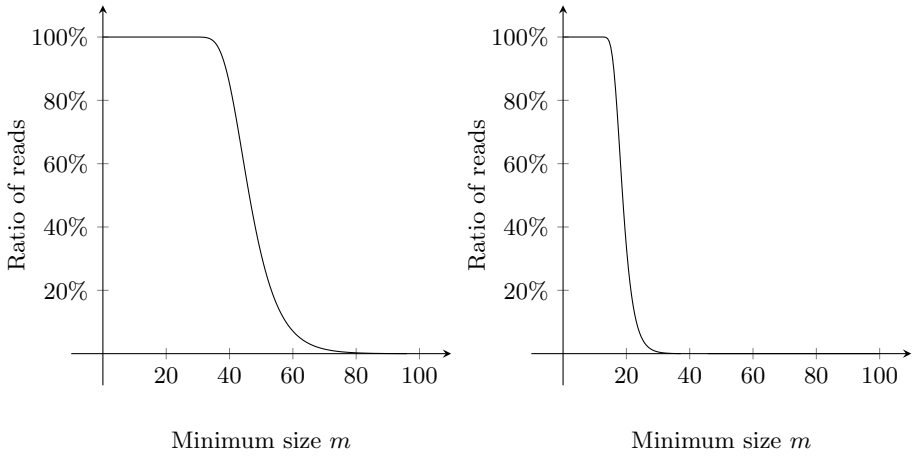

Fig. 3. Expected percentage of reads of size 10000 that contain at least one exact region of size $k \geq m$, for reads with 15% errors (*left*) and 35% errors (*right*).

3.3 Applications

During the local chaining step from section 2.3 one can apply the results of section 3.1 to decide whether a local mapping is plausible or not. For each mapping the coverage by exact regions can easily be computed by counting seed sizes. The resulting number can then be compared to the expected coverage that can be obtained from section 3.1. If there is a significant deviation in either direction, the local mapping gets a lower rating.
When computing mappings it is required to have at least 1 seed available, hence the results from section 3.2 propose good upper bounds for the minimum length of seeds, depending on the read size and error rates. To a certain extent this result can also be used to estimate the probability of a read containing several exact regions of a minimal size. If a read of size n contains a MEM of size $k \geq m$, then this MEM divides the read in two pieces, one of size n' and the other of (approx.) size $n - n'$. This approximation of the piece-sizes is made since typically k is significantly smaller than n, and k is not known a priori. The conditional probability of the read containing a second MEM of size larger than m then becomes $1 - (1 - P(n', p, m))(1 - P(n - n', p, m))$, with P as in (5). Since n' depends on the read, it is a priori not known and integrating over n' is required. The distribution of the size of n' can be approximated by the uniform distribution on $\{0, \ldots, n\}$, and because of symmetry this leads to the following estimate of the a priori probability of a read of size n containing at least 2 exact regions with size larger than m:

$$P(n, p, m)\frac{2}{n}\sum_{n'=0}^{n/2}\Big(1 - \big(1 - P(n', p, m)\big)\big(1 - P(n - n', p, m)\big)\Big)\,. \tag{6}$$

Equation (6) can in a similar fashion be extended to multiple seeds, possibly of different minimal sizes. However one should be careful when using (6) and other extensions of (5), since the approximation made by $P(n, p, m)$ becomes less accurate when n decreases.

4 Results

4.1 Data

Datasets were simulated from reference genomes of varying size, two bacterial genomes: N. meningitidis and A. hydrophila; and one eukaryotic genome: D. melanogaster (fruit fly). For all genomes tests are performed on a perfect de Bruijn graph built from the reference genome. For the bacterial genomes short reads are simulated from which additional de Bruijn graphs are built. Illumina paired-end reads of length 100 were simulated with ART [26] with 100x coverage. PacBio reads of average length 10000 were simulated with pbsim [27] and 10x coverage, with 15% errors distributed as 60% insertions, 30% deletions and 10% substitutions. Real Illumina and PacBio datasets were used for E. coli. For D. melanogaster real PacBio data was mapped on the de Bruin graph built from the reference genome. The sources of the data are specified in table 1.

Table 1. The data sets and reference genomes. The D. melanogaster reference genome can be accessed on http://www.fruitfly.org/sequence/release5genomic.shtml.

N. meningitidis	Reference genome NC_003116.
A. hydrophila	Reference genome NC_008570.
D. melanogaster	Reference genome Release 5. PacBio data from http://datasets.pacb.com.s3.amazonaws.com/2014/Drosophila/reads/list.html.
E. coli	Reference genome NC_000913. Illumina data accession number ERR022075. PacBio data from https://github.com/PacificBiosciences/DevNet/wiki/E coli K12 MG1655 Hybrid Assembly

4.2 Parameters

LoRDEC was run with $k = 19$ for the bacterial data sets, as suggested in [6]. For the larger fruit fly the values $k = 20 \ldots 23$ were tested. The best results were obtained for $k = 21$ and these results have been included in the results tables. For Jabba the de Bruijn graphs were built with $k = 31$ and the minimum MEM size was 20.

4.3 Evaluation metrics

In [6] it is demonstrated that LoRDEC performs better than both LSC [13] and PacBioToCA [14]. Hence, Jabba is only compared to LoRDEC. By using simulated data, the corrected read can be aligned to the original sequence from which the read was simulated. This way a multiple alignment of the original read, the corrected read and the genomic region is created. In this alignment each position is analyzed separately, in order to obtain a confusion matrix as follows:

- True Positive, an erroneous position was corrected.
- False Positive, a new error was introduced at a correct position.
- True Negative, a correct position remains unchanged.
- False Negative, an erroneous position remains unchanged.

To interpret this confusion matrix the following statistics are computed:

- Sensitivity $= TP/(TP + FN)$. This expresses the relative amount of errors that were corrected.
- Specificity $= TN/(TN + FP)$. This expresses the relative amount of correct positions that were recognized as such.
- Precision $= TP/(TP + FP)$. This expresses how reliable a correction is, if one is made.
- Gain $= (TP - FP)/(TP + FN)$. This expresses the quality of the corrected reads compared to the original read.

For real data the reads are aligned to the reference genome with BLASR [28] and the identity of the mapping is computed. All experiments were run on dual-socket octa-core Intel Xeon Sandy Bridge computing nodes at 2.6 GHz and 64 GB of memory. The runtimes and memory usage are measured using the standard Linux time command. The runtime includes only the actual mapping of long reads and does not include the generation and correction of the de Bruijn graph.

Table 2. Results on simulated data for LoRDEC and Jabba. The subscript $_p$ indicates the usage of a perfect de Bruijn graph built from the reference genome. The absence of the subscript indicates the usage of a de Bruijn graph built from simulated short reads. Both real time and CPU time are averages over all reads, with 16 threads in parallel.

	Sensitivity	Specificity	Precision	Gain	Real Time	CPU Time	Memory
Neisseria meningitidis							
LoRDEC	95.6 %	99.0 %	94.4 %	89.9 %	266.5 ms	1732.5 ms	63 MB
LoRDEC$_p$	98.7 %	99.3 %	96.0 %	94.6 %	141.7 ms	921.1 ms	58 MB
Jabba	92.3 %	99.1 %	94.4 %	86.9 %	126.1 ms	1596.5 ms	126 MB
Jabba$_p$	92.3 %	99.1 %	94.5 %	87.0 %	125.0 ms	1583.3 ms	115 MB
Aeromonas hydrophila							
LoRDEC	99.2 %	99.7 %	98.4 %	97.6 %	84.1 ms	1093.4 ms	75 MB
LoRDEC$_p$	99.8 %	99.9 %	99.1 %	99.0 %	70.5 ms	597.3 ms	46 MB
Jabba	95.7 %	99.4 %	96.4 %	92.1 %	180.0 ms	2315.2 ms	161 MB
Jabba$_p$	95.3 %	99.4 %	96.1 %	91.4 %	178.5 ms	2311.3 ms	168 MB
Drosophila melanogaster							
LoRDEC$_p$	91.6 %	98.5 %	91.1 %	82.7 %	435.1 ms	2828.6 ms	538 MB
Jabba$_p$	93.9 %	99.2 %	95.3 %	89.3 %	246.4 ms	2285.7 ms	1181 MB

4.4 Evaluation

Table 2 shows the results for LoRDEC and Jabba as they were run on each of the simulated data sets. The results on the bacterial genomes suggest that a corrected de Bruijn graph yields comparable results to using a perfect de Bruijn Graph based on a reference genome. For the bacteria, LoRDEC performs better than Jabba, since the MEMs contain the same information as k-mers, this is most likely due to shortcomings in the chaining algorithm in Jabba. A noticeable drop in the performance of LoRDEC can be observed when using the frequency-based de Bruijn graph compared to the perfect de Bruijn graph.
The performance of Jabba does not drop when moving from the bacterial genomes to the fruit fly. On the other hand, our evaluation of LoRDEC shows that LoRDEC obtains a lower gain for this larger genome, to the point where Jabba outperforms it. It can be seen in table 2 that Jabba is slower than LoRDEC on A. hydrophila, but faster on the other two genomes, and that Jabba consistently requires 2 to 4 times more memory than LoRDEC.

Table 3. Results on real data for LoRDEC and Jabba, as obtained by mapping the reads to the reference genome.

	Original Identity	LoRDEC Identity	LoRDEC Gain	Jabba Identity	Jabba Gain
Escherichia coli	83.8 %	98.6 %	91.9 %	98.5 %	90.8 %
Drosophila melanogaster	86.7 %	96.6 %	74.3 %	98.0 %	85.3 %

The results for the real data can be found in table 3. The higher performance of Jabba on D. melanogaster when compared to LoRDEC might be explained by the use of MEMs. In LoRDEC the seed size and the k-mer size of the graph are identical. Since the seed size must be kept relatively low as shown in section 3.2, the de Bruijn graph has to be built with a small value of k, leading to a tangled de Bruijn graph. This is expected to become more apparent for larger and more complex genomes. When using MEMs however, the seed size and k-mer size are independent of each other and optimal values of k can be chosen to construct the de Bruijn graph.

Acknowledgments. The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, the Hercules Foundation and the Flemish Government – department EWI. We acknowledge the support of Ghent University (Multidisciplinary Research Partnership "Bioinformatics: From Nucleotides to Networks").

References

1. Salzberg, S.L. et al.: GAGE: a critical evaluation of genome assemblies and assembly algorithms. Genome Res., 22, 557–567 (2012)
2. Yang, X., Chockalingam, S.P., Aluru, S.: A survey of error-correction methods for next-generation sequencing. Brief. Bioinform., 14(1), 56–66 (2013)
3. Kelley, D.R., Schatz, M.C., Salzberg, S.L.: Quake: quality-aware detection and correction of sequencing errors. Genome Biol., 11:R116 (2010)
4. Greenfield, P. et al.: Blue: correcting sequencing errors using consensus and context. Bioinformatics, 30(19), 2723–2732 (2014)
5. Zerbino, D.R., Birney, E.: Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res., 18, 821–829 (2008)
6. Salmela, L., Rivals, E.: LoRDEC: accurate and efficient long read error correction. Bioinformatics, 30(24), 3506–3514 (2014)
7. Schröder, J. et al.: SHREC: a short-read error correction method. Bioinformatics, 25(17), 2157–2163 (2009)
8. Ilie, L., Fazayeli, F., Ilie, S.: HiTEC: accurate error correction in high-throughput sequencing data. Bioinformatics, 27(3), 295–302 (2011)
9. Salmela, L., Schröder, J.: Correcting errors in short reads by multiple alignments. Bioinformatics, 27(11), 1455–1461 (2011)
10. Myers, G.: Efficient local alignment discovery amongst noisy long reads. Algorithms in Bioinformatics, LNCS, 8701, 52–67 (2014)

11. Berlin, K. et al.: Assembling large genomes with single-molecule sequencing and locality sensitive hashing. Nat. Biotech. 33, 623–630 (2015).
12. Boetzer, M., Pirovano, W.: SSPACE-LongRead: scaffolding bacterial draft genomes using long read sequence information. BMC bioinform. 15(1), 211 (2014).
13. Au, K. F. et al.: Improving PacBio Long Read Accuracy by Short Read Alignment. PLoS ONE 7(10), e46679 (2012)
14. Koren S. et al.: Hybrid error correction and de novo assembly of single-molecule sequencing reads. Nat. Biotechnol. 30, 693–700 (2012)
15. Hackl, T. et al.: proovread: large-scale high-accuracy PacBio correction through iterative short read consensus. Bioinformatics, 30(21), 3004–3011 (2014)
16. Liu, Y., Schmidt, B.: Long read alignment based on maximal exact match seeds. Bioinformatics, 28(18), i318-i324 (2012)
17. Vyverman, M. et al.: A long fragment aligner called ALFALFA. BMC Bioinformatics, 16, 159 (2015)
18. Li, H.: Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv:1303.3997 [q-bio.GN] (2013)
19. Li, H., Durbin, R.: Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics, 26(5), 589–595 (2009)
20. Langmead, B., Salzberg, S.L.: Fast gapped-read alignment with Bowtie 2. Nat. Methods, 9(4), 357–359 (2012)
21. Vyverman, M. et al.: essaMEM: finding maximal exact matches using enhanced sparse suffix arrays. Bioinformatics 29(6), 802–804 (2013)
22. Zhao, M. et al.: SSW Library: An SIMD Smith-Waterman C/C++ Library for Use in Genomic Applications. PLoS ONE, 8 (2013)
23. Arratia, R., Gordon, L., Waterman, M.S.: An extreme value theory for sequence matching. The Annals of Statistics, 14(3), 971–993 (1986)
24. Gordon, L., Schilling, M. F., Waterman, M. S.: An extreme value theory for longest head runs. Zeitschrift fur Wahrscheinlichkeitstheories verwandt Gebeite (Probability Theory and Related Fields), 72, 279–287 (1986)
25. Schilling, M.F.: The Surprising Predictability of Long Runs, Math. Assoc. of Am., 85(2), 141–149 (2012)
26. Huang, W. et al.: ART: a next-generation sequencing read simulator. Bioinformatics, 28(4), 593–594 (2012)
27. Ono, Y., Asai, K., Hamada, M.: PBSIM: PacBio reads simulator–toward accurate genome assembly. Bioinformatics, 29(1), 119–121 (2013)
28. Chaisson M.J., Tesler G.: Mapping single molecule sequencing reads using Basic Local Alignment with Successive Refinement (BLASR): Theory and Application, BMC Bioinformatics, 13238 (2012)